Dynamic Markov Trees Based Benchmark Compression for Power, Temperature, and Reliability Estimation of CMPs

Cristinel Ababei
Milwaukee WI, USA, Feb. 2015

***Abstract*—In this study, we propose the use of dynamic Markov trees to develop innovative benchmark compression techniques and to naturally model the spatial characteristics of thread scheduling. These models could enable a new class of benchmark compression approaches that effectively capture the effect of scheduling on the estimation of power, thermal profile, and lifetime reliability – metrics which are increasingly relevant in network-on-chip based chip multiprocessors (CMPs) that have become the main trend in processor development.**

I. Full System Simulation of CMPs

Full system simulation to evaluate the functionality and performance of architectural innovations of chip multiprocessors is crucial to computer architects. The main challenge is that benchmarks (e.g., applications) require very long simulation times due to the large number of instructions – in the order of billions or more – that must be simulated. For example, detailed simulation of single or multicore processors is typically thousands of times slower than the simulated hardware. One minute of execution in real time can correspond to days or weeks of simulation time.

Aside from using algorithm parallelization, techniques that use statistical simulation [1], simpoints [2] or statistical sampling with checkpointing [3] can reduce the total simulation effort required to estimate the performance of such applications by orders of magnitude. Although such techniques have received a level of interest that makes them a research area in its own right, the issue of faster and more accurate simulation is still largely an open problem. In addition, it is not obvious how these techniques account for the impact of thread scheduling especially in the increasingly popular chip multiprocessors. This is important due to primarily two reasons. First, threads scheduling directly affects the power consumption of different cores, thereby the overall thermal profiles as well as the lifetime reliability. Imagine for example a single threaded application being run on a four core processor where the single thread is always run on the same core as opposed to being switched between all four cores. The thermal and thereby the lifetime reliability in these two scenarios are two completely different pictures. In addition, especially in CMPs that use increasingly popular networks-on-chip as the communication medium, traffic through the network is also affected by scheduling. The physical distance between cores which run threads that exchange data impacts the amount of data traffic or cache coherency messaging in the networks.

This represents the main motivation for the application compression technique proposed in this study. Our innovative method is based on dynamic Markov trees that enable: 1) benchmark compression such that simulation time can be shortened thousands of times and 2) natural modeling of spatial characteristics of thread scheduling, thereby facilitating accurate estimation of metrics of interest.

II. Dynamic Markov Trees

The idea of benchmark compression is to somehow capture the essential characteristics and inherent variations of the original benchmark into a convenient abstraction that can then be utilized to *replay* the benchmark in a much shorter version such that estimations of target metrics (e.g., power, temperature, lifetime reliability) result into values that would be obtained as if the original, much longer, benchmark would be simulated. Inspired by [4], we introduce dynamic Markov trees (DMTs) as an abstraction to capture essential spatial behavior and to enable benchmark compression. A graphical illustration of a DMT is shown in Fig.1.

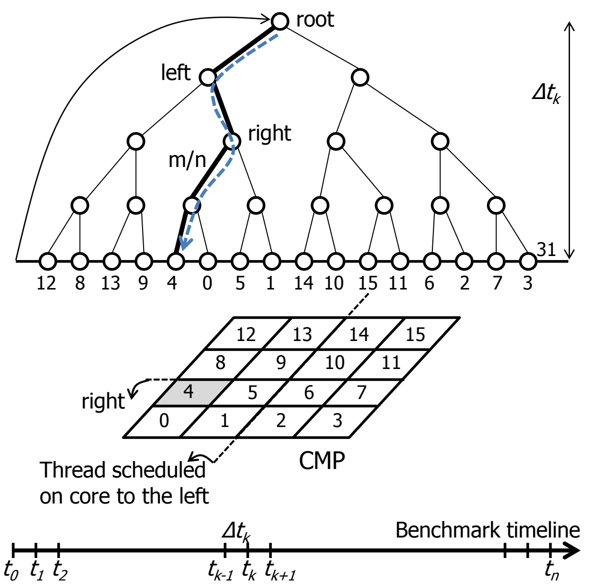

Figure 1. Illustration of dynamic Markov tree construction for a chip multiprocessor with 16 cores. Counters associated with arcs are incremented during each Δt_k along paths that indicate the core on which the thread is scheduled. In this example, the thread is scheduled on the core in tile 4 of the CMP. At the end of all n iterations the probability of an arc is the ratio m/n, where m is the counter value.

A DMT is essentially a binary tree with a root node and a number of intermediate node layers that depends on the size of the CMP. It is constructed as follows. Starting from the root node, which represents the top-level CMP, we create two child nodes that correspond to the two halves of the CMP architecture. The two halves represent a first partitioning level of the CMP. This partitioning with new nodes creation process is continued hierarchically until each partition contains one tile only.

Arcs of the tree are labeled with probabilities that denote the chance of going from the parent node to each of the child nodes depending on whether a given thread is scheduled to a core of a tile located in either of the two halves. These probabilities are calculated during an iterative process whose total number of iterations, n, equals the number of time intervals that we split the entire run (denoted as timeline in Fig.1) of a given benchmark. During each iteration, we walk the tree from the root node down toward the leaf nodes to indicate the core on which a thread is scheduled during this time interval, Δt_k. Then, we go back to the root node and start another walk that corresponds to the next time interval, Δt_{k+1}. During these DMT *construction* walks, each arc along the walking path from the root node to the leaf node has its counter incremented. At the end of all n iterations, arc probabilities are given by the ratio between the final values of these counters and the total number of iterations, n.

Once the DTM is constructed, benchmark compression can be achieved by performing a new set of walks on the tree with a number of iterations n' that is much smaller than the number of iterations n done during the DMT construction: $n' << n$. During each iteration of the new walk, a path from the root node to a leaf node is followed *randomly* as dictated by the probabilities associated with each arc. In this way, the compressed benchmark (shown abstractly in Fig.2) is generated iteratively in only n' iterations.

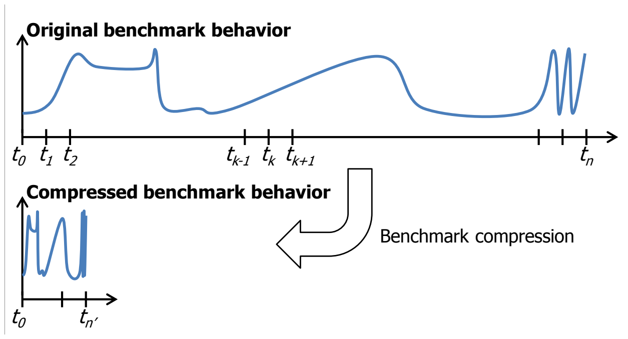
Original benchmark behavior
t_0 t_1 t_2 t_{k-1} t_k t_{k+1} t_n
Compressed benchmark behavior
Benchmark compression
t_0 $t_{n'}$

Figure 2. Illustration of benchmark compression.

The compressed benchmark can then be utilized for full system simulations. For any architectural changes (e.g, core microarchitecture, dynamic voltage and frequency scaling schemes, etc.) that need to be investigated, the compressed benchmark can be run to estimate figures of merit such as CMP lifetime reliability thousands of times faster.

The DMT model is straightforwardly extended to the case when the benchmark is multithreaded or the top-level application is a multicase or composite situation (i.e., multiple benchmarks are run concurrently). In this case, the DTM can be enhanced by maintaining lists attached to each node and arc of the DTM. These lists would have as many entries as the number of threads.

III. Discussion

This study raises rather than answers questions. Because understanding benchmark behavior is at the foundation of computer architecture and software optimization, we believe that the proposed benchmark compression approach opens up a series of ideas that are exiting to investigate.

For example, one needs to find the value of n' that offers an optimal compromise between achieved simulation speed up and statistical confidence of the estimated figures of merit. Also, an interesting exercise is to think of how the DTM model can capture also temporal not only spatial relationships between threads scheduled on different cores. We would like to also explore how the proposed DTM can be utilized to construct enhanced versions of simulation techniques that use statistical simulation, simpoints, or statistical sampling with checkpointing [1]–[3].

IV. Conclusion

We proposed a novel benchmark compression technique based on dynamic Markov trees for CMP research. The proposed model can effectively capture the effect of scheduling on the estimation of power, thermal profile, and lifetime reliability. It can effectively speed up full system simulations and therefore be tremendously helpful to computer architects.

References

[1] S. Nussbaum and J. E. Smith, “Modeling superscalar processors via statistical simulation," *Int. Conf. on Parallel Architectures and Compilation Techniques*, 2001.

[2] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, “Automatically characterizing large scale program behavior," *Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, 2002.

[3] R.E. Wunderlich, T.F. Wenisch, B. Falsafi, and J.C. Hoe, “Smarts: accelerating microarchitecture simulation via rigorous statistical sampling," *Int. Symp. on Computer Architecture*, 2003.

[4] R. Marculescu, D. Marculescu, and M. Pedram, “Vector compaction using dynamic Markov models," *IEICE Trans. Fundamentals*, vol. E80-A, no. IO, p. 1924-1933, Oct. 1997.

[5] N. Binkert, B. Beckmann, G. Black, S.K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D.R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewall, M. Shoaib, N. Vaish, M.D. Hill, and D.A. Wood, “The gem5 simulator," *ACM SIGARCH Computer Architecture News Archive*, vol. 39, no. 2, May 2011.